LAB 7

STAT 28

March 2, 2017

Welcome to lab 7! In this lab, you will:

- Learn how to do loess regression.
- Explore some visualization methods for datasets with categorical variables.

Loess: Local Polynomial Regression Fitting

Read the data.

```
data_science <- read.csv("data_science.csv")
# convert string to date object
data_science$week <- as.Date(data_science$week, "%Y-%m-%d")
# create a numeric column representing the time
data_science$time <- as.numeric(data_science$week)
data_science$time <- data_science$time - data_science$time[1] + 1
```

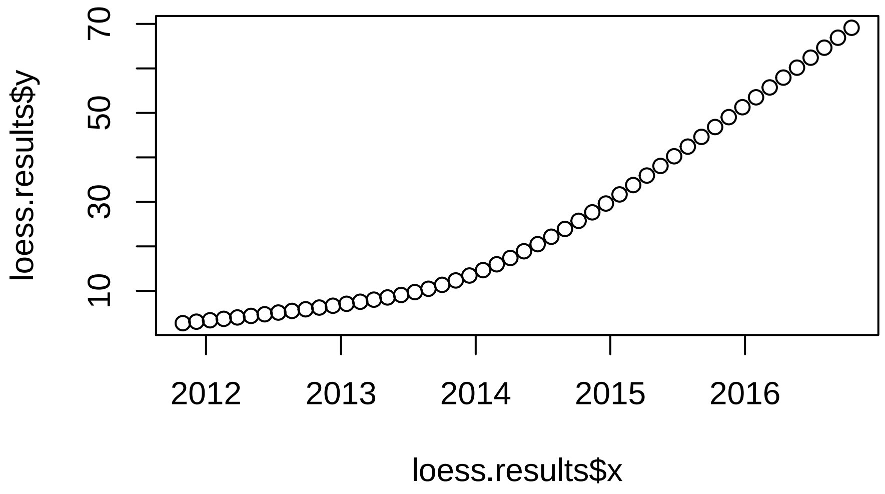
There are several options in R for loess fitting.

`loess.smooth` returns a list with the smoothed data coordinates:

```
loess.results <- loess.smooth(x=data_science$week,y=data_science$r)
loess.results
```

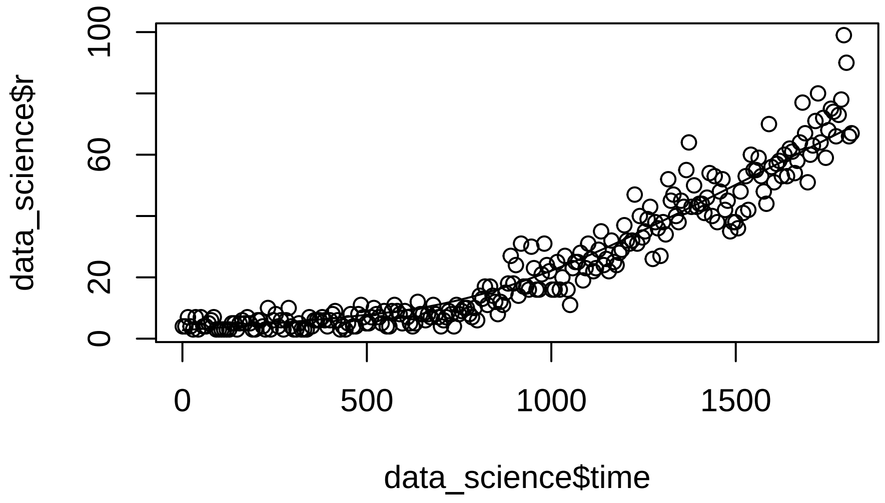
```
## $x
##  [1] "2011-10-30" "2011-12-06" "2012-01-12" "2012-02-18" "2012-03-26"
##  [6] "2012-05-02" "2012-06-08" "2012-07-15" "2012-08-21" "2012-09-27"
## [11] "2012-11-03" "2012-12-10" "2013-01-16" "2013-02-22" "2013-03-31"
## [16] "2013-05-07" "2013-06-13" "2013-07-20" "2013-08-26" "2013-10-02"
## [21] "2013-11-08" "2013-12-15" "2014-01-21" "2014-02-27" "2014-04-05"
## [26] "2014-05-12" "2014-06-18" "2014-07-25" "2014-08-31" "2014-10-07"
## [31] "2014-11-13" "2014-12-20" "2015-01-26" "2015-03-04" "2015-04-10"
## [36] "2015-05-17" "2015-06-23" "2015-07-30" "2015-09-05" "2015-10-12"
## [41] "2015-11-18" "2015-12-25" "2016-01-31" "2016-03-08" "2016-04-14"
## [46] "2016-05-21" "2016-06-27" "2016-08-03" "2016-09-09" "2016-10-16"
##
## $y
##  [1]  2.736377  3.072615  3.392789  3.708313  4.030601  4.371064  4.741115
##  [8]  5.123584  5.494673  5.865706  6.248098  6.653264  7.092621  7.554054
## [15]  8.022257  8.522621  9.080914  9.722903 10.474354 11.353310 12.346029
## [22] 13.448366 14.657861 15.972053 17.388485 18.901316 20.494740 22.166603
## [29] 23.916373 25.743519 27.647511 29.627548 31.675587 33.776878 35.916586
## [36] 38.079874 40.251906 42.420154 44.608423 46.820078 49.046064 51.277326
## [43] 53.504812 55.719529 57.936788 60.169443 62.411485 64.656900 66.899678
## [50] 69.133807
```

```
plot(loess.results$x, loess.results$y)
```



`loess.smooth` is very convenience in the sense of plotting. Actually, there is also `scatter.smooth`, which plot the scatter plot and smoothed fit with just one line of code. (The disadvantage of `scatter.smooth` is that there is no argument to change the line color. And it is just for plotting, no coordinates are returned.)

```
scatter.smooth(x=data_science$time,y=data_science$r)
```



However, `loess.smooth` only returns the smoothed data points. To do prediction using the loess model, we need function `loess`, which has similar usage with the `lm` function. Notice the default smoothing parameter for `loess.smooth`(span = 2/3, degree = 1) and `loess`(span = 0.75, degree = 2) is different.

```r
loess.fitted <- loess(r~time, data = data_science)
summary(loess.fitted)
```

```
## Call:
## loess(formula = r ~ time, data = data_science)
##
## Number of Observations: 260
## Equivalent Number of Parameters: 4.35
## Residual Standard Error: 5.464
## Trace of smoother matrix: 4.73  (exact)
##
## Control settings:
##   span     :  0.75
##   degree   :  2
##   family   :  gaussian
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```

```r
predict(loess.fitted, data.frame(time = c(1000, 1500)))
```

```
## [1] 20.73465 50.04658
```

Exercise 1.

Follow the steps below to create a plot:

- (1) Plot the scatter plot of `week` versus `r` and `week` versus `python` with red and blue. You may use high-level function `plot` for the first group, and use low-level function `points` to add the other group of points. You may adjust the point size using argument `cex = 0.5`. Do not forget to add the labels and title.
- (2) Plot the loess smoothing line on top with red and blue. You can use `loess.smooth` and low-level plot function `lines`. You may want to the argument `lwd = 2` to increase the line width.
- (3) Add the legend to the plot.

Eventually, your plot will look like the following:

```r
# Insert your code for plotting here
```

Multivariate Data visualization - Online news popularity

This dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. This dataset contains 49 variables for each news post, such as

- `weekday`: days of week. Mon, Tue, Wed, etc.
- `channel`: channel. Tech, Entertainment, Business, etc.
- `shares`: number of shares.
- `num_imgs`: Number of images.
- `num_videos`: Number of videos.
- `n_tokens_title`: : Number of words in the title.
- `num_hrefs`: Number of links

Read in data.

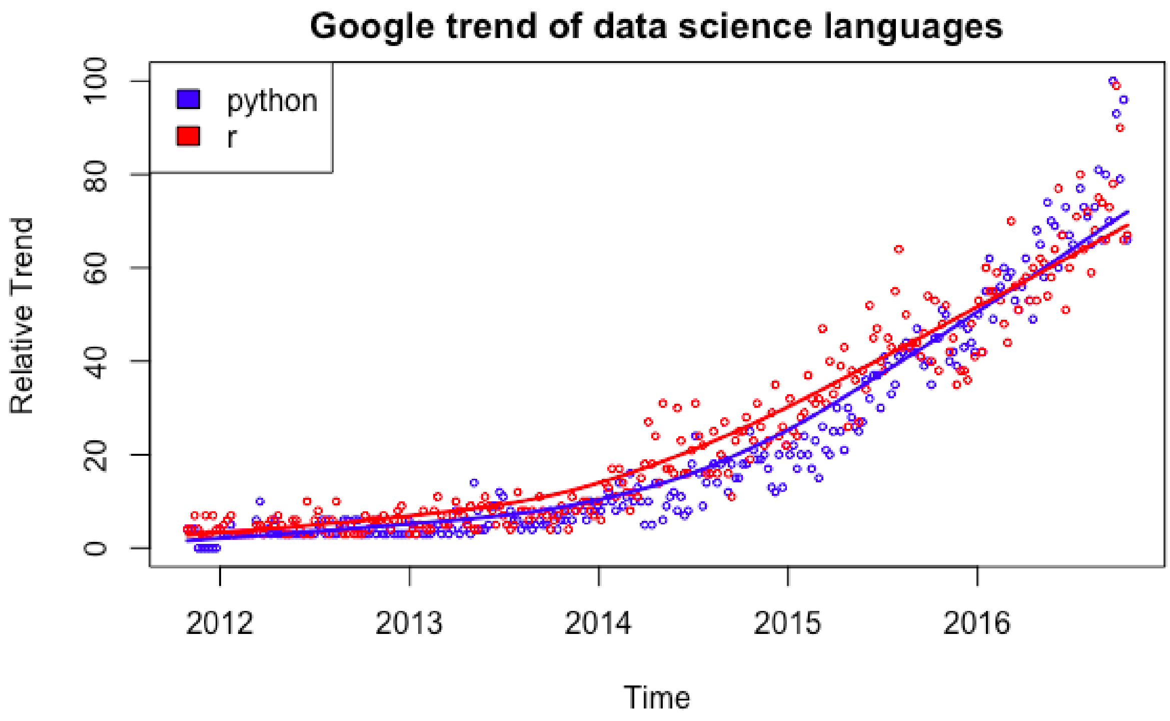


Figure 1:

```r
popul <- read.csv("OnlineNewsPopularity.csv")
```

Exercise 2

(a) Make the bar plots for `weekdays` and `channels` separately side by side using `barplot` and `table` function. Rotate the x axis labels using argument `las = 2` in `barplot`. (HINT: use `par(frow...)`)

```r
# Insert your code for plotting here
```

(b) Create a contingency table between `weekdays` and `channels`. Use barplot to visualize the relationship between two categorical variables.

```r
# Insert your code for plotting here
```

(c) Use the contingency table you created in (b) to get separate bar plots for days of the week. (use `beside = TRUE`)

```r
# Insert your code for plotting here
```

Exercise 3

Followed is a subset of the news popularity dataset.

```r
vars <- c("weekday", "channel", "shares", "num_imgs", "num_videos", "n_tokens_title", "num_hrefs")
sample.idx <- sample(nrow(popul), 2000)
subset <- popul[sample.idx, vars]
```

(a) Plot the pairs plot using the function `pairs` for `subset`.

```r
# Insert your code for plotting here
```

(b) Plot the pairs plot using the function `gpairs` in the `gpairs` package for `subset`. Which function is more friendly for categorical variables?

```r
library(gpairs)
# Insert your code for plotting here
```

Exercise Extra (not required)

(a) Plot the alluvial plot for `weekday` and `channel`.

```r
library(alluvial)
# Insert your code for plotting here
```

(b) Plot the mosaicplot plot for `weekday` and `channel`.

```r
# Insert your code for plotting here
```